SGCWEBSOCKETS · TECHNICAL DOCUMENT

CEX.io API

CEX.io WebSocket and REST client — public and authenticated channels for the CEX.io exchange.

eSeGeCe

WWW.ESEGECE.COM

Overview

WebSocket API allows getting real-time notifications without sending extra requests, making it a faster way to obtain data from the exchange

At a glance

COMPONENT CLASS `TsgcWSAPI_Cex`	STANDARDS / SPEC **CEX.io REST API**
TRANSPORTS **TCP, TLS**	PLATFORMS **Windows, macOS, Linux, iOS, Android**
FRAMEWORKS **VCL, FireMonkey, Lazarus / FPC, .NET**	EDITION **Standard / Professional / Enterprise**

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	CEX.io REST API · CEX.io WebSocket API
Component class	`TsgcWSAPI_Cex` (unit `sgcWebSocket_API_Cex`)
Frameworks	VCL, FireMonkey, Lazarus / FPC, .NET
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`Client`	Published or public property used to configure or query the component.
`OnCexSubscribed`	Published or public property used to configure or query the component.
`OnCexAuthenticated`	Published or public property used to configure or query the component.
`OnConnect`	Published or public property used to configure or query the component.
`OnCexConnect`	Published or public property used to configure or query the component.
`OnCexMessage`	Published or public property used to configure or query the component.
`OnCexError`	Published or public property used to configure or query the component.
`OnCexDisconnecting`	Published or public property used to configure or query the component.
`OnCexDisconnect`	Published or public property used to configure or query the component.
`OnDisconnect`	Published or public property used to configure or query the component.

Main methods

The principal public methods exposed by the component.

`SubscribeOrderBook()`	Public procedure exposed by the component.
`UnSubscribeOrderBook()`	Public procedure exposed by the component.

`CancelOrderRequest()`	Public procedure exposed by the component.
`Ping()`	Public procedure exposed by the component.
`Authenticate()`	Public procedure exposed by the component.
`SubscribeTickers()`	Public procedure exposed by the component.
`UnSubscribeTickers()`	Public procedure exposed by the component.
`SubscribePair()`	Public procedure exposed by the component.
`UnSubscribePair()`	Public procedure exposed by the component.
`SubscribeChart()`	Public procedure exposed by the component.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Binance | Connect WebSocket API** configuration sourced from the online help.

> **About this scenario.** In order to connect to Binance WebSocket API, just create a new Binance API client and attach to TsgcWebSocketClient.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oBinance := TsgcWSAPI_Binance.Create(nil);
oBinance.Client := oClient;
oClient.Active := True;
```

C++ Builder

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(NULL);
oBinance->Client = oClient;
oClient->Active = true;
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Binance oBinance = new TsgcWSAPI_Binance();
oBinance.Client = oClient;
oClient.Active = true;
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · Bitmex | Connect WebSocket API

In order to connect to Bitmex WebSocket API, just create a new Bitmex API client and attach to TsgcWebSocketClient.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oBitmex := TsgcWSAPI_Bitmex.Create(nil);
oBitmex .Client := oClient;
oClient.Active := True;
```

C++ Builder

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex→Client = oClient;
oClient→Active = true;
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex.Client = oClient;
oClient.Active = true;
```

2 · Coinbase | Connect WebSocket API

In order to connect to Coinbase WebSocket API, just create a new Coinbase API client and attach to TsgcWebSocketClient. See below an example:

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oCoinbase := TsgcWSAPI_Coinbase.Create(nil);
oCoinbase.Client := oClient;
oClient.Active := True;
```

C++ Builder

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase();
oCoinbase->Client = oClient;
oClient->Active = true;
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Coinbase oCoinbase = new TsgcWSAPI_Coinbase();
oCoinbase.Client = oClient;
oClient.Active = true;
```

3 · Kucoin | Connect WebSocket API

In order to connect to Kucoin WebSocket API, just create a new Kucoin API client and attach to TsgcWebSocketClient.

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oKucoin := TsgcWSAPI_Kucoin.Create(nil);
oKucoin.Client := oClient;
oClient.Active := True;
```

C++ Builder

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin *oKucoin = new TsgcWSAPI_Kucoin();
oKucoin->Client = oClient;
oClient->Active = true;
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Kucoin oKucoin = new TsgcWSAPI_Kucoin();
oKucoin.Client = oClient;
oClient.Active = true;
```

4 · Kucoin | Futures Connect WebSocket API

In order to connect to Kucoin WebSocket API, just create a new Kucoin API client and attach to TsgcWebSocketClient.

```
Delphi (VCL / FireMonkey)
```

```
oClient := TsgcWebSocketClient.Create(nil);
oKucoin := TsgcWSAPI_Kucoin_Futures.Create(nil);
oKucoin.Client := oClient;
oClient.Active := True;
```

```
C++ Builder
```

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient();
</code><code class="delphi">TsgcWSAPI_Kucoin_Futures </code><code class="cpp">*oKucoin = new </c
oKucoin->Client = oClient;
oClient->Active = true;
```

```
.NET (C#)
```

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
</code><code class="delphi">TsgcWSAPI_Kucoin_Futures </code><code class="csharp">oKucoin = new <
oKucoin.Client = oClient;
oClient.Active = true;
```

5 · Binance | Subscribe WebSocket Channel

Binance offers a variety of channels where you can subscribe to get real-time updates of market data, orders... Find below a sample of how to subscribe to a Ticker:

```
Delphi (VCL / FireMonkey)
```

```
oClient := TsgcWebSocketClient.Create(nil);
oBinance := TsgcWSAPI_Binance.Create(nil);
oBinance.Client := oClient;
oBinance.SubscribeTicker('bnbbtc');

procedure OnMessage(Connection: TsgcWSConnection; const aText: string);
begin
// here you will receive the ticker updates
end;
```

C++ Builder

```
TsgcWebSocketClient *oClient = new TsgcWebSocketClient(NULL);
TsgcWSAPI_Binance *oBinance = new TsgcWSAPI_Binance(NULL);
oBinance->Client = oClient;
oBinance->SubscribeTicker("bnbbtc");

void OnMessage(TsgcWSConnection *Connection, const string aText)
{
// here you will receive the ticker updates
}
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Binance oBinance = new TsgcWSAPI_Binance();
oBinance.Client = oClient;
oBinance.SubscribeTicker("bnbbtc");

void OnMessage(TsgcWSConnection Connection, const string aText)
{
// here you will receive the ticker updates
}
```

6 · Bitmex | Subscribe WebSocket Channel

Bitmex offers a variety of channels where you can subscribe to get real-time updates of market data, orders... Find below a sample of how subscribe to a Trade Channel:

Delphi (VCL / FireMonkey)

```
oClient := TsgcWebSocketClient.Create(nil);
oBitmex := TsgcWSAPI_Bitmex.Create(nil);
oBitmex.Client := oClient;
oBitmex.Subscribe(btmTrade, 'XBTUSD');
procedure OnBitmexMessage(Sender: TObject; const aTopic: TwsBitmexTopics; const aMessage: string
begin
// here you will receive the trade updates
end;
```

C++ Builder

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex->Client = oClient;
oBitmex->Subscribe(btmTrade, "XBTUSD");
void OnBitmexMessage(Sender: TObject; const aTopic: TwsBitmexTopics; const aMessage: string)
{
// here you will receive the trade updates
}
```

.NET (C#)

```
TsgcWebSocketClient oClient = new TsgcWebSocketClient();
TsgcWSAPI_Bitmex oBitmex = new TsgcWSAPI_Bitmex();
oBitmex.Client = oClient;
oBitmex.Subscribe(btmTrade, "xbtusd");
void OnBitmexMessage(Sender: TObject; const aTopic: TwsBitmexTopics; const aMessage: string)
{
// here you will receive the tradeupdates
}
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — CEX.io REST API	docs.cex.io/
Primary standard / spec — CEX.io WebSocket API	docs.cex.io/#websocket-api
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/APIs/API/API_Cex.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\05.Crypto\04.Cex`
.NET demo project (in the sgcWebSockets package)	`.net\demos\05.Crypto\04.Cex`
Component page	www.esegece.com/products/websockets/apis/cex-io/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the CEX.io API component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.